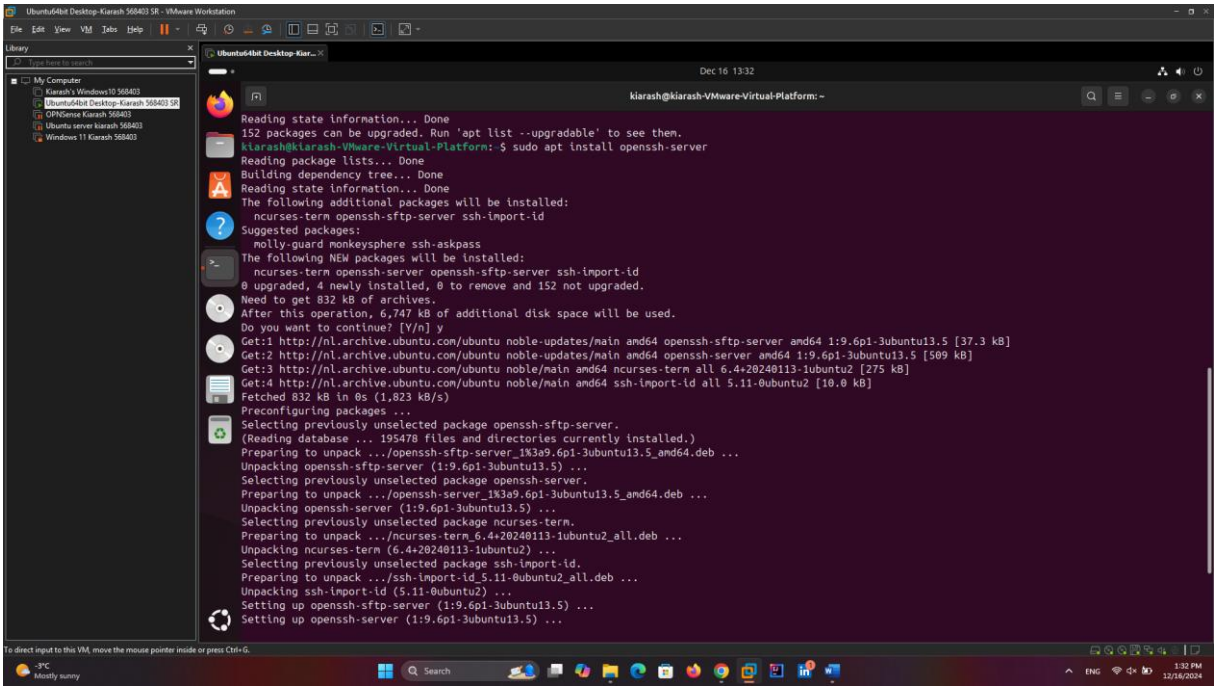
Template Week 6 – Networking

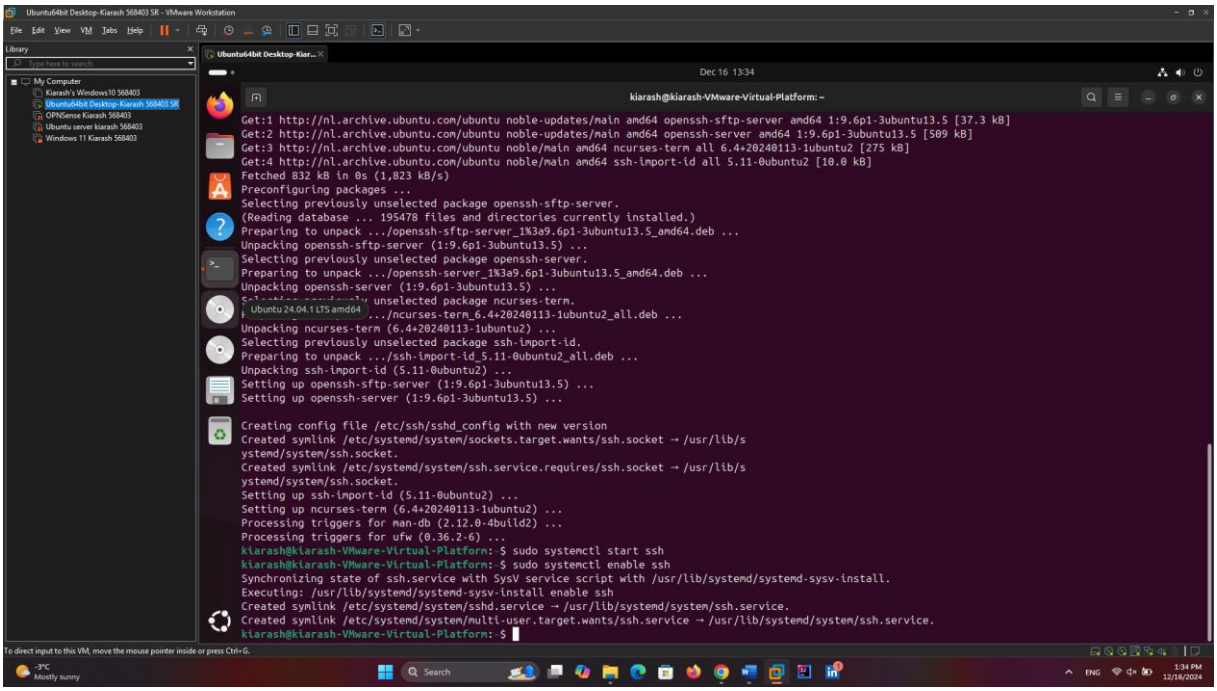
Student number: Kiarash Delavar-568403

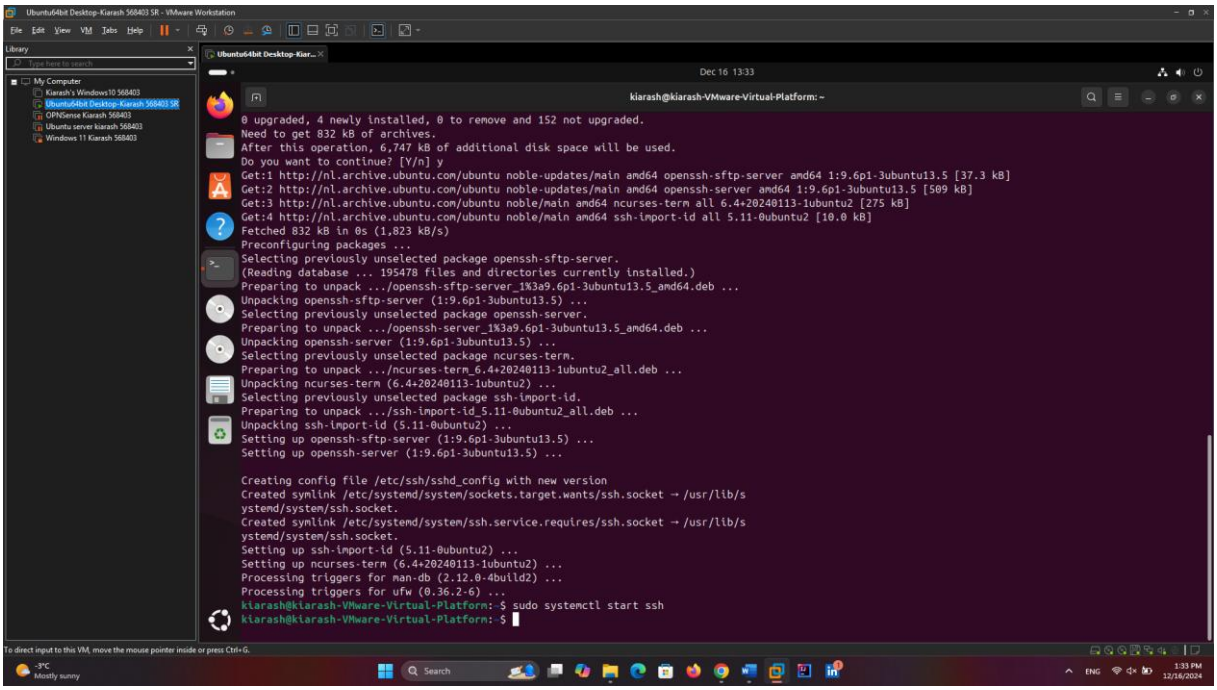
Assignment 6.1: Working from home

Screenshot installation openssh-server:

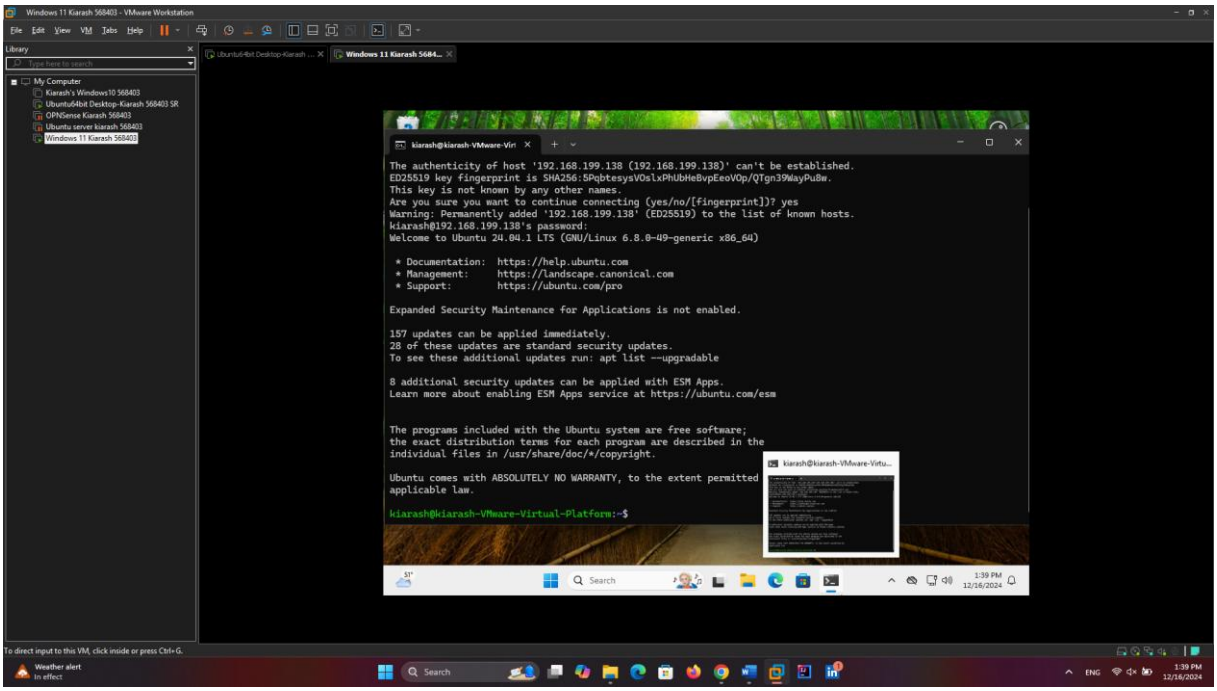


Screenshot successful SSH command execution:

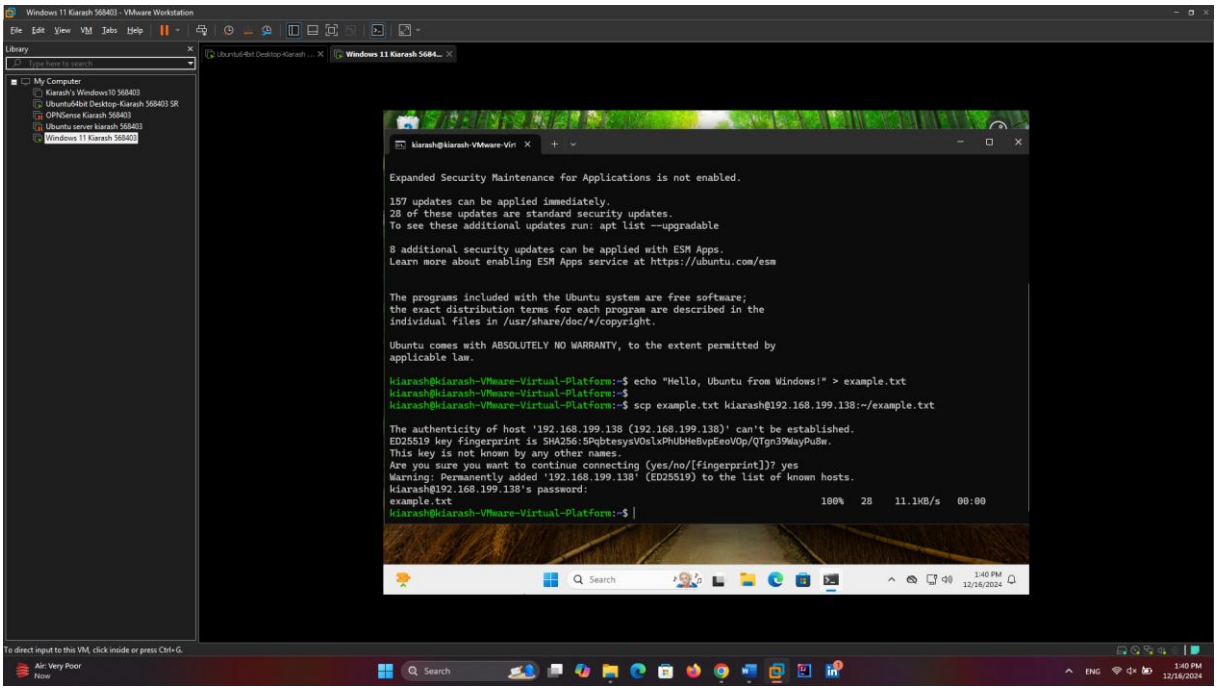




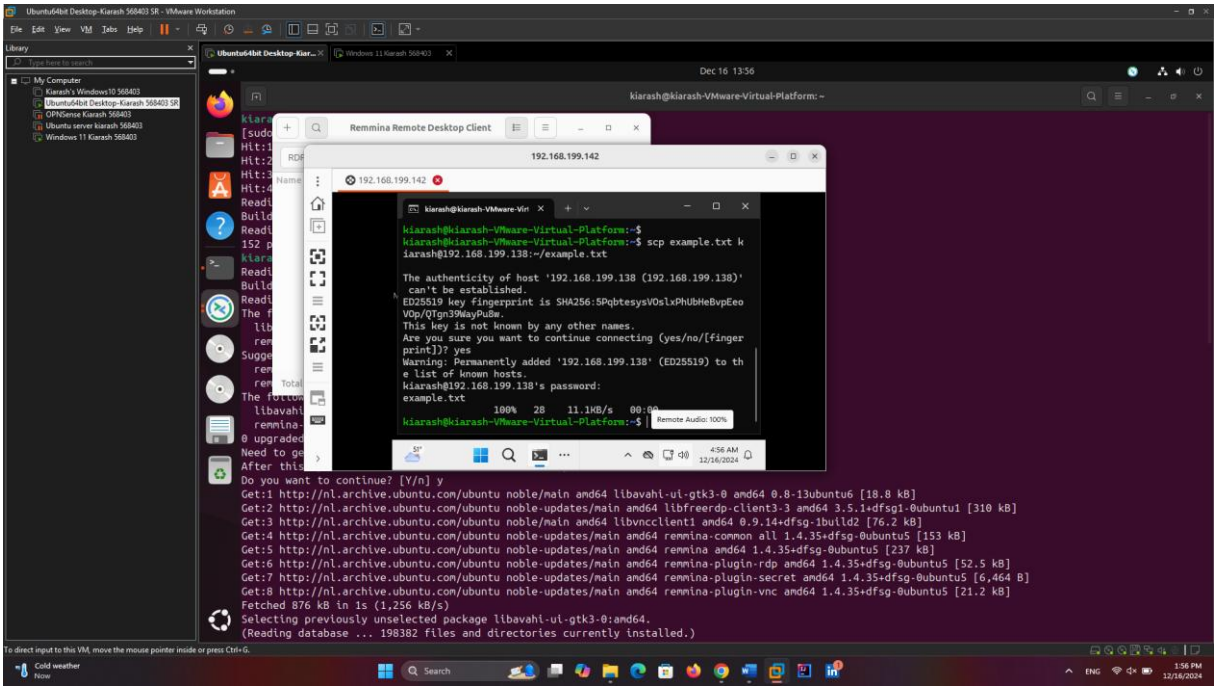
On Windows 11 VM:



Screenshot successful execution SCP command:

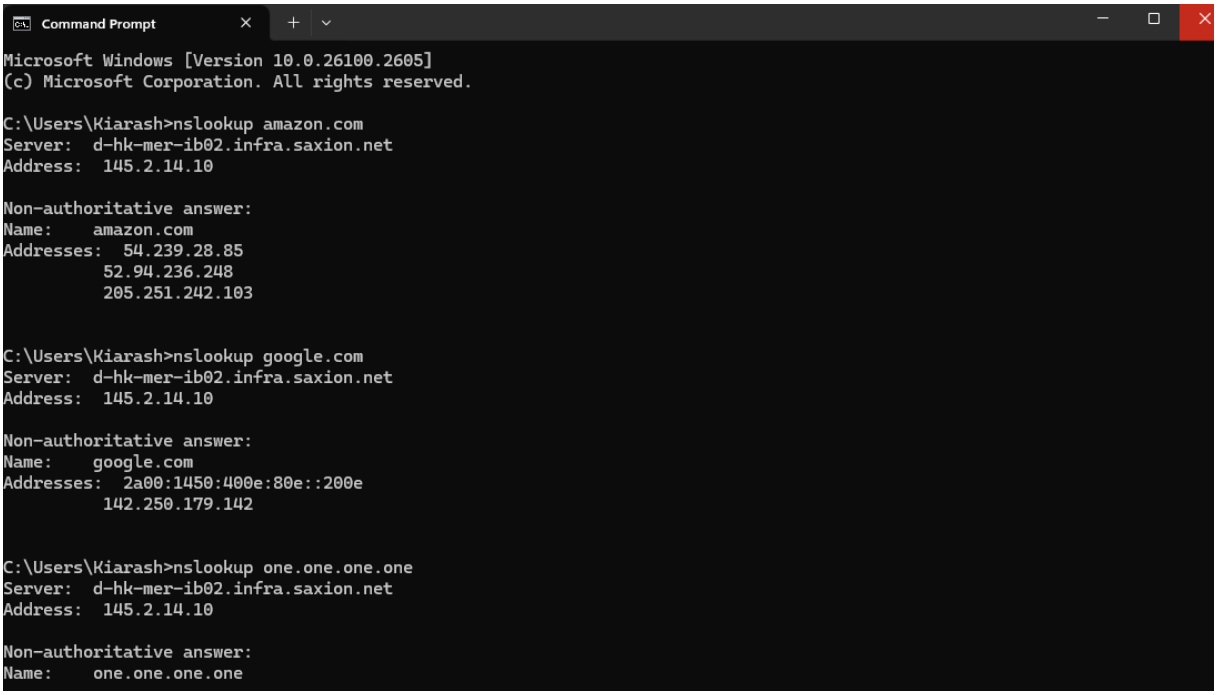


Screenshot remmina:

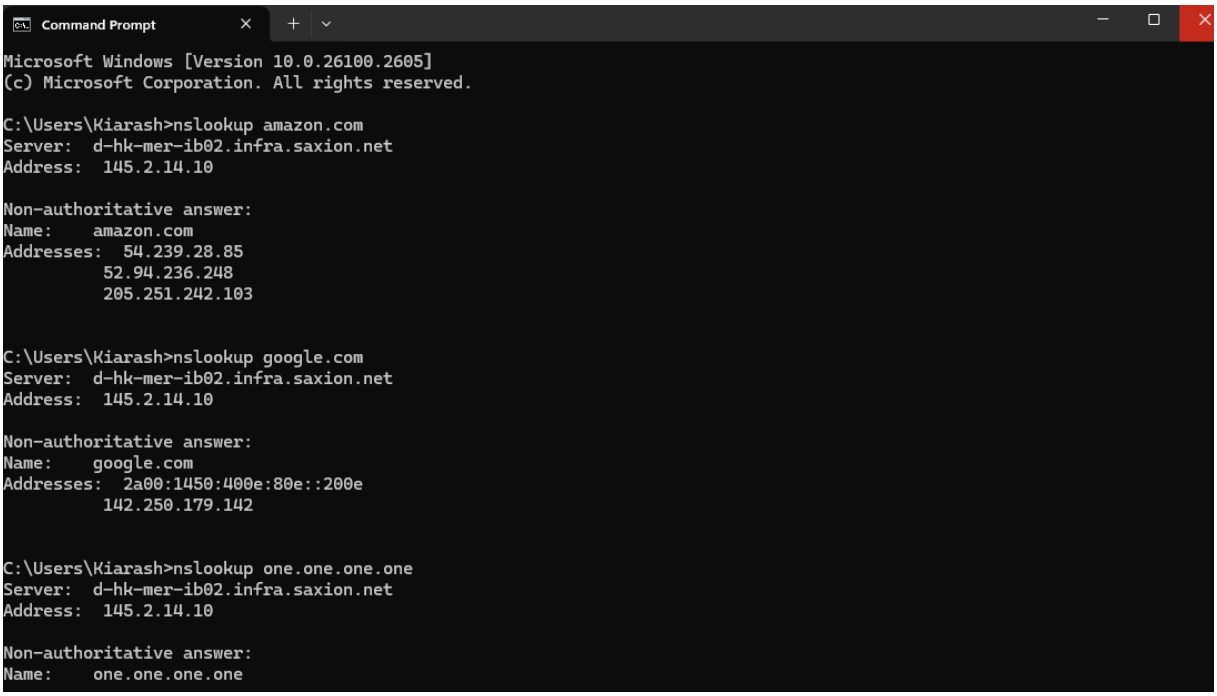


Assignment 6.2: IP addresses websites

Relevant screenshots nslookup command:



Screenshot website visit via IP address:



Assignment 6.3: subnetting

How many IP addresses are in this network configuration 192.168.110.128/25?

+ 128 IP addresses (2^ (32-25) = 2^7 = 128)

What is the usable IP range to hand out to the connected computers?

+ Is 126 (128-2). Exclude the **network address** (first address in the range) and the **broadcast address** (last address in the range).

Check your two previous answers with this calculator:
https://www.calculator.net/ip-subnet-calculator.html

Explain the above calculation in your own words:

1- **Total IP Addresses:**
The subnet 192.168.110.128/25 means there are a total of **128 IP addresses** in this network. This includes all the addresses from 192.168.110.128 to 192.168.110.255.

2- **Usable IP Addresses**:
Out of these 128 addresses, the first one (192.168.110.128) is the **network address**, and the last one (192.168.110.255) is the **broadcast address**. These two addresses cannot be assigned to devices.
That leaves **126 usable addresses**, which range from 192.168.110.129 to 192.168.110.254.

3- **Subnet Mask**:
The subnet mask for /25 is 255.255.255.128. This means the first 25 bits of the IP are fixed for the network, while the remaining 7 bits are used for devices (hosts) in the network.

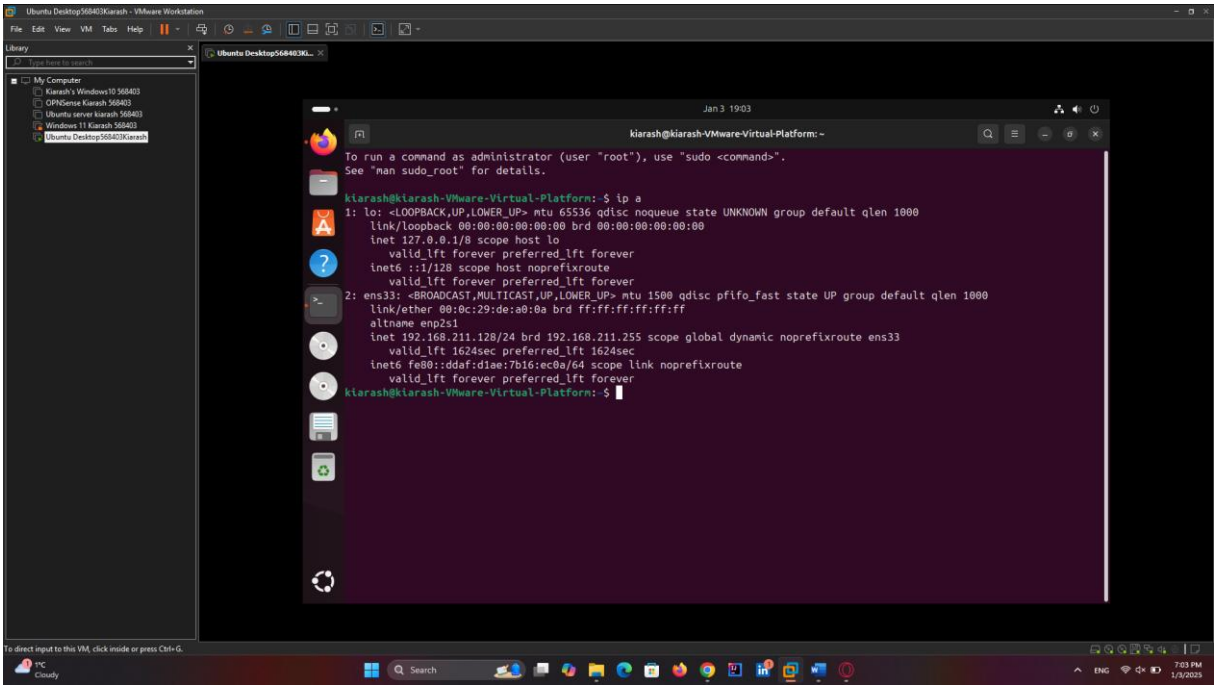
4- **How the Calculator Confirms This**:

The subnet calculator shows:

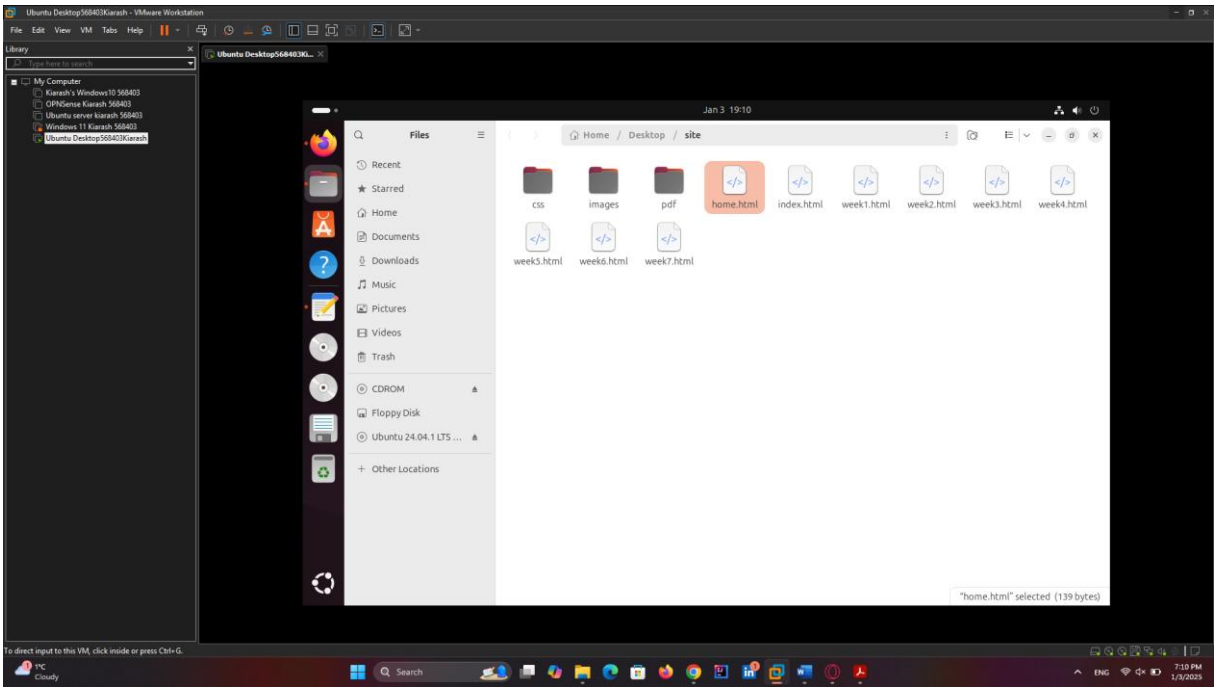
- Total IPs: 128
- Usable range: 192.168.110.129 to 192.168.110.254
- Usable hosts: 126
- Network address: 192.168.110.128
- Broadcast address: 192.168.110.255

Assignment 6.4: HTML

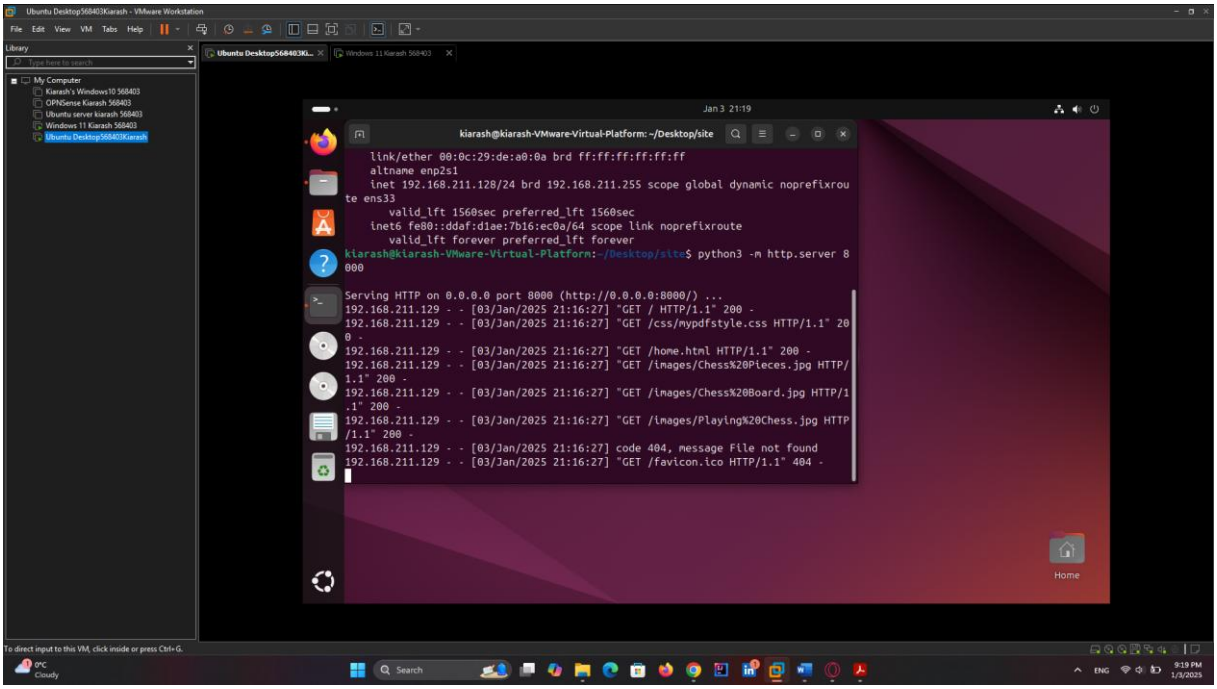
Screenshot IP address Ubuntu VM:



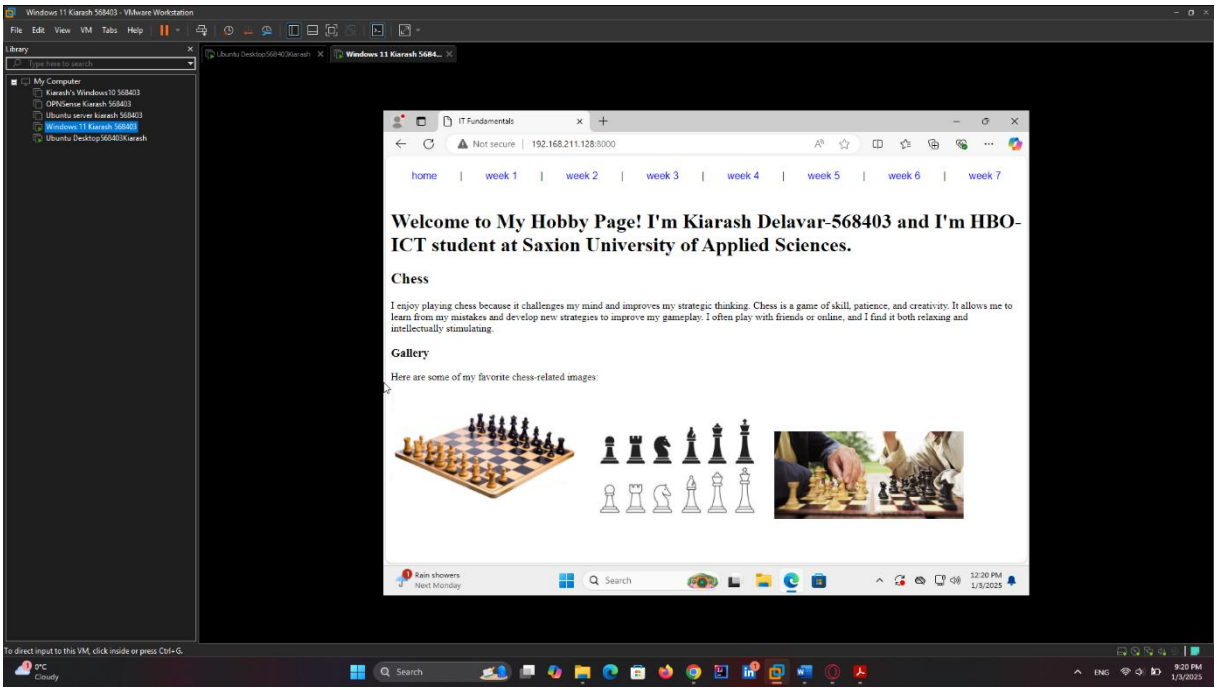
Screenshot of Site directory contents:



Screenshot python3 webserver command:



Screenshot web browser visits your site: (Windows 11 VM-Microsoft Edge)



Bonus point assignment – week 6

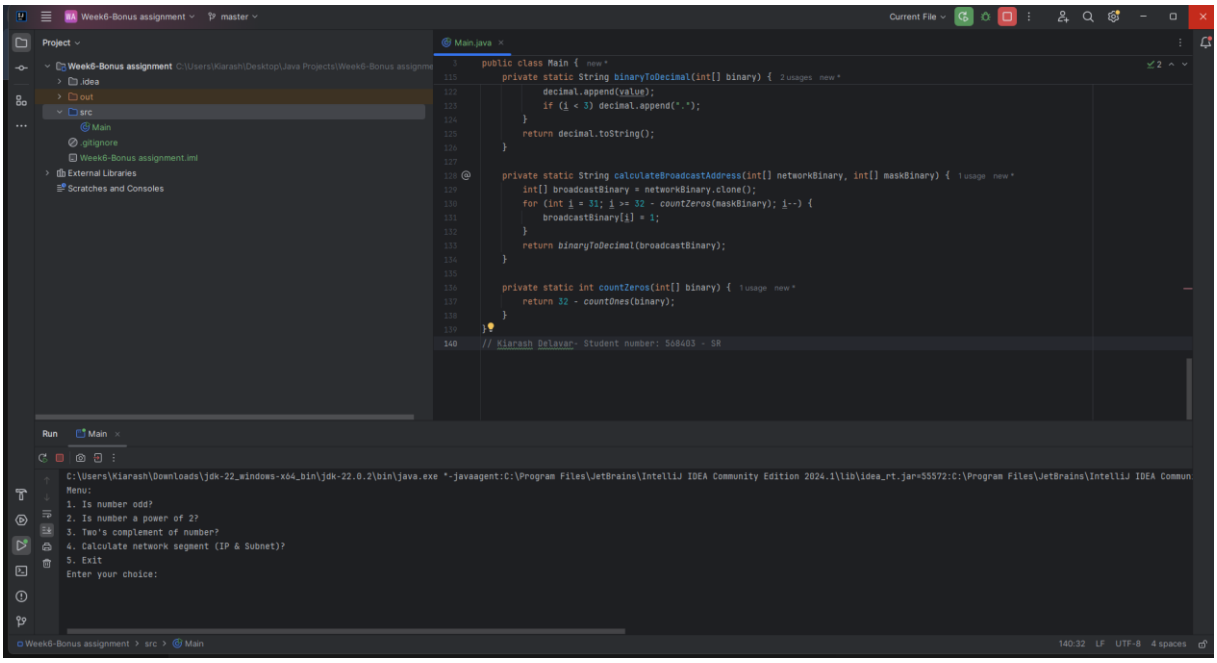
Remember that bitwise java application you've made in week 2? Expand that application so that you can also calculate a network segment as explained in the PowerPoint slides of week 6. Use the bitwise & AND operator. You need to be able to input two Strings. An IP address and a subnet.

IP: 192.168.1.100 and subnet: 255.255.255.224 for /27

```
Example: 192.168.1.100/27
Calculate the network segment
IP Address:   11000000.10101000.00000001.01100100
Subnet Mask:  11111111.11111111.11111111.11100000
-------------------------------------------------
Network Addr: 11000000.10101000.00000001.01100000

This gives 192.168.1.96 in decimal as the network address.
For a /27 subnet, each segment (or subnet) has 32 IP addresses (2⁵).
The range of this network segment is from 192.168.1.96 to 192.168.1.127.
```

Paste source code here, with a screenshot of a working application:



Please See the next page!

```
3. Two's complement of number?
4. Calculate network segment (IP & Subnet)?
5. Exit
Enter your choice: 4
Enter IP Address (e.g., 192.168.1.100): 192.168.1.102
Enter Subnet Mask (e.g., 255.255.255.224 or /27): /27
Network Address: 192.168.1.96
Total IPs in Subnet: 32
IP Range: 192.168.1.96 to 192.168.1.127
```

Source code here:

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int choice;
        do {
            System.out.println("Menu:");
            System.out.println("1. Is number odd?");
            System.out.println("2. Is number a power of 2?");
            System.out.println("3. Two's complement of number?");
            System.out.println("4. Calculate network segment (IP & Subnet)?");
            System.out.println("5. Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    System.out.print("Enter a number: ");
                    int numOdd = scanner.nextInt();
                    System.out.println("Is the number odd? " + isOdd(numOdd));
                    break;
                case 2:
                    System.out.print("Enter a number: ");
                    int numPower = scanner.nextInt();
                    System.out.println("Is the number a power of 2? " + isPowerOfTwo(numPower));
                    break;
                case 3:
                    System.out.print("Enter a number: ");
                    int numTwosComplement = scanner.nextInt();
                    System.out.println("Two's complement of the number: " +
twosComplement(numTwosComplement));
                    break;
                case 4:
```

```
            scanner.nextLine(); // Consume the newline character
            System.out.print("Enter IP Address (e.g., 192.168.1.100): ");
            String ipAddress = scanner.nextLine();
            System.out.print("Enter Subnet Mask (e.g., 255.255.255.224 or /27): ");
            String subnetMask = scanner.nextLine();
            calculateNetworkSegment(ipAddress, subnetMask);
            break;
         case 5:
            System.out.println("Exiting...");
            break;
         default:
            System.out.println("Invalid choice. Please try again.");
      }
      System.out.println();
   } while (choice != 5);

   scanner.close();
}

private static boolean isOdd(int number) {
   return (number & 1) == 1;
}

private static boolean isPowerOfTwo(int number) {
   return number > 0 && (number & (number - 1)) == 0;
}

private static int twosComplement(int number) {
   return ~number + 1;
}

private static void calculateNetworkSegment(String ipAddress, String subnetMask) {
   int[] ipBinary = convertToBinary(ipAddress);
   int[] maskBinary = subnetMask.startsWith("/")
         ? cidrToBinary(subnetMask)
         : convertToBinary(subnetMask);

   int[] networkAddressBinary = new int[32];
   for (int i = 0; i < 32; i++) {
      networkAddressBinary[i] = ipBinary[i] & maskBinary[i];
   }

   String networkAddress = binaryToDecimal(networkAddressBinary);

   int totalIPs = 1 << (32 - countOnes(maskBinary));
   String broadcastAddress = calculateBroadcastAddress(networkAddressBinary, maskBinary);

   System.out.println("Network Address: " + networkAddress);
```

```
        System.out.println("Total IPs in Subnet: " + totalIPs);
        System.out.println("IP Range: " + networkAddress + " to " + broadcastAddress);
    }

    private static int[] convertToBinary(String ip) {
        String[] parts = ip.split("\\.");
        int[] binary = new int[32];
        for (int i = 0; i < 4; i++) {
            int value = Integer.parseInt(parts[i]);
            for (int j = 0; j < 8; j++) {
                binary[i * 8 + (7 - j)] = (value >> j) & 1;
            }
        }
        return binary;
    }

    private static int[] cidrToBinary(String cidr) {
        int prefixLength = Integer.parseInt(cidr.replace("/", ""));
        int[] binary = new int[32];
        for (int i = 0; i < prefixLength; i++) {
            binary[i] = 1;
        }
        return binary;
    }

    private static int countOnes(int[] binary) {
        int count = 0;
        for (int bit : binary) {
            if (bit == 1) count++;
        }
        return count;
    }

    private static String binaryToDecimal(int[] binary) {
        StringBuilder decimal = new StringBuilder();
        for (int i = 0; i < 4; i++) {
            int value = 0;
            for (int j = 0; j < 8; j++) {
                value += binary[i * 8 + j] << (7 - j);
            }
            decimal.append(value);
            if (i < 3) decimal.append(".");
        }
        return decimal.toString();
    }

    private static String calculateBroadcastAddress(int[] networkBinary, int[] maskBinary) {
        int[] broadcastBinary = networkBinary.clone();
```

```
        for (int i = 31; i >= 32 - countZeros(maskBinary); i--) {
            broadcastBinary[i] = 1;
        }
        return binaryToDecimal(broadcastBinary);
    }

    private static int countZeros(int[] binary) {
        return 32 - countOnes(binary);
    }
}
```

Ready? Save this file and export it as a pdf file with the name: **week6.pdf**